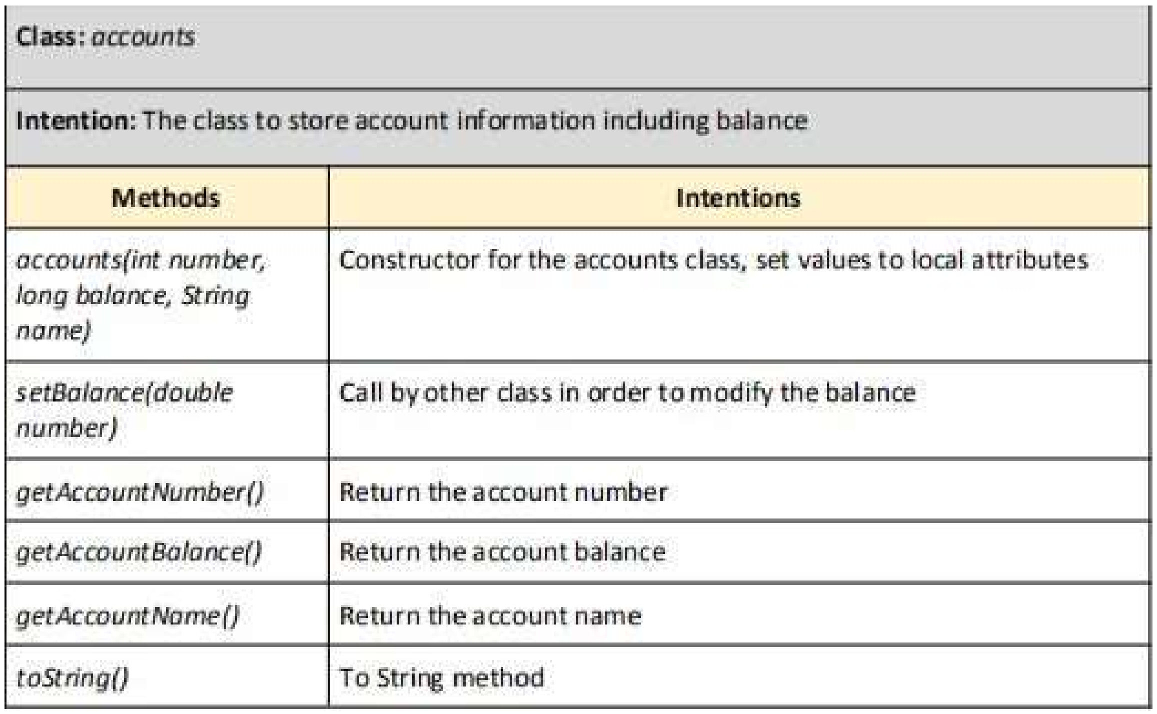
<u>Group Name: DevsSkipQA</u>
Zilong Zhang - 20062161
Runqing Zang - 20065628
Derek Huang - 20022672
Kalsa Yu - 20073438

<u>Design Document (back office)</u>

A design document, giving the overall structure of your solution, showing the ***classes*** and ***methods*** as a diagram or table, with ***a brief (one sentence) description of the intention*** of each.

Class: backend	
Intention: backend system of the banking system	
Methods	**Intentions**
readOldMaster(String fileName)	Read the old master account file
readTrans(String fileName)	Read new transaction summary file(s)
updateMaster(String[] updateFile)	update master account file with new transaction summary file(s)
updateValid(String[] updateFile)	update the valid account list with new transaction summary file(s)
reverseSort()	Sort the master account in descending order
newValidList()	Create the new valid account list based on old master account + new transaction summary files
checkMaster(ArrayList<String> result)	Check constraints of the new master account file
writeMaster(String path)	write new master account file
writeValid(String path)	write new valid account list
main(String args[])	Main function of the class



Class: *accounts*	
Intention: The class to store account information including balance	
Methods	**Intentions**
accounts(int number, long balance, String name)	Constructor for the accounts class, set values to local attributes
setBalance(double number)	Call by other class in order to modify the balance
getAccountNumber()	Return the account number
getAccountBalance()	Return the account balance
getAccountName()	Return the account name
toString()	To String method

In particular, the backend class will take at least two inputs, where the first input is the old master account file, the rest of inputs will be transaction summary file(s). Through the prototype, if all constraints are met, the old master account file will be updated using the input transaction summary file(s) and sorted in reverse order according to account numbers. Also, a new valid list file will be created using the input transaction summary file(s). The program will then write the new master account file and new valid account list.

<u>Table of Results are in Test table</u>

<u>Detailed Failure Report for our test runs</u>

Test number	Test Purpose	How output failed	Code error	How it was fixed
All test	All testing	All test cases involves	Error was in code.	Fixed by changing the

cases involves using the reverse sort function	purposes	the use of the *reverse sort* function does not produce the expected output.	Inequality operators are written in the opposite direction.	sign (> to <) in *reverseSort* function
Multiple tests	All testing purposes	Some test cases such as wrong transaction summary format are failed to be checked with the backend	Logical issue in code structure and test designing	We have modify/deleted such test cases since the backend assumes the front end input files are correct and make sure the output format are correct. Such test cases should be re-visited in Integration.
Multiple tests	All testing purposes	Master account write format should be 000 for empty balance but appeared to be 0.0	Error was in code.	Fixed by changing zero balance to 000 string and let other accounts with balance to case to be integer
WR1T1	Check if master account file holds a line that exceeding 47 characters.	Instead of giving the “Exceeding length” error, program gives “Success”	Error was in test.	We found out that if the input is a number, it will automatically cast to scientific notation. We then changed the test code to alphabetic characters. (since large amounts in scientific notation should be allowed)
WR5T5	Check If account name are legal	Legal name widraw operation should be successful but failed	Error was in test. Did not give enough balance to the account.	Modified the test code so that account balance is enough for widraw.
WR6T1	Check if monetary	Should print “Error”	Error in code	Fixed by adding

	is less than 3 digits	but did not.	Did not check if the account number begin with 0	constraints to the string at char(0).
All Test cases	Check if program should terminate upon error or continue to work	Failure as display by maven	Error in code.	Found that boolean value is not working properly. Use String as the end check instead.

createacct **(white box input partition testing)**

constraints for files	Requirement	Partition	Test No.	content in transaction summary file	content in old master accounts file	expected new master file	expected new valid account list	expected terminal output	Table of Results
constraints for master accounts file	each line in master file has at most 47 characters(plus newline)	more than 47 characters	R1T1	NEW 1234568 000 0000000 name	1234567 1000 nameeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee	Error	Error	Exceeding length	As expected
		less than or equal to 47 characters	R1T2	NEW 1234568 000 0000000 name	1234567 1000 kalsa	1234568 000 name 1234567 1000 kalsa	1234568 1234567	Success	As expected
	Items are separated by exactly one space	items are separated by more than one space	R2T1	NEW 1234568 000 0000000 name	1234567 1000 kalsa	Error	Error	Not seprated by space	As expected
		no space between items	R2T2	NEW 1234568 000 0000000 name	12345671000kalsa	Error	Error	Not seprated by space	As expected
		items separated by one space	R2T3	NEW 1234568 000 0000000 name	1234567 1000 kalsa	1234568 000 name 1234567 000 kalsa	1234568 1234567	Success	As expected
	no account should ever have negative balance	account with negative balance	R4T1	NEW 1234566 000 0000000 name	1234567 -1000 kalsa	Error	Error	Account balance cannot be less than 0	As expected
		account with positive balance	R4T2	NEW 1234566 000 0000000 name	1234567 1000 kalsa	1234567 1000 kalsa 1234566 000 name	1234567 1234566	Success	As expected
constraints for transaction summary file	a created account must have a new, unused account number	create account with a used account number	R5T1	NEW 1234567 000 0000000 name	1234567 1000 kalsa	Error	Error	Created account must have a new unused account number	As expected
		create accountg with a unused account number	R5T2	NEW 1234566 000 0000000 name	1234567 1000 kalsa	1234567 1000 kalsa 1234566 000 name	1234567 1234566	Success	As expected
	account numbers are always exactly seven digits, not beginning with 0	account number more than 7 digits	R6T1	NEW 12345678 000 0000000 name	1234567 1000 kalsa	Error	Error	Account number begin with zero or not seven digits	As expected
		account number beginning with 0	R6T2	NEW 0123456 000 0000000 name	1234567 1000 kalsa	Error	Error	Account number begin with zero or not seven digits	As expected
		legal account number	R6T3	NEW 1234566 000 0000000 name	1234567 1000 kalsa	1234567 1000 kalsa 1234566 000 name	1234567 1234566	Success	As expected
		legal monetary	R7T3	NEW 1234566 000 0000000 name	1234567 1000 kalsa	1234567 1000 kalsa 1234566 000 name	1234567 1234566	Success	As expected
	account names are between 3 and 30 alphanumeric characters, possibly including spaces, but not beginning or ending with a space	name less than 3 digits	R8T1	NEW 1234566 000 0000000 na	1234567 1000 kalsa	Error	Error	Name not between 3 and 30 chars or begin or end with space	As expected
		name more than 30 digits	R8T2	NEW 1234566 000 0000000 namenamenamenamenamenamenamename	1234567 1000 kalsa	Error	Error	Name not between 3 and 30 chars or begin or end with space	As expected
		legal name	R8T5	NEW 1234566 000 0000000 name	1234567 1000 kalsa	1234567 1000 kalsa 1234566 000 name	1234567 1234566	Success	As expected

withdraw transactions **(white box output partition testing)**

constraints for files	Requirement	Output Partition	Test No.	content in transaction summary file	content in old master accounts file	expected new master file	expected new valid account list	expected terminal output	Table of Results
constraints for master accounts file	each line in master file has at most 47 characters(plus WDRline)	more than 47 characters	R1T1	WDR 1234567 1000 0000000 name	1234567 1000000000000000000000000000000000000000 name	Error	Error	Exceeding length	As expected
		less than or equal to 47 characters	R1T2	WDR 1234567 1000 0000000 name	1234567 1000 name	1234567 000 name	1234567	Success	As expected
	Items are separated by exactly one space	items separated by one space	R2T1	WDR 1234567 1000 0000000 name	1234567 1000 name	1234567 000 name	1234567	Success	As expected
	The Master file must always kept in descending order by account number	withdraw account with a larger account number	R3T1	WDR 1234567 1000 0000000 name	1234568 1000 kalsa 1234567 1000 name	1234568 1000 kalsa 1234567 000 name	1234568 1234567	Success	As expected
	no account should ever have negative balance	account with negative balance	R4T1	WDR 1234567 1000 0000000 name	1234567 -1000 name	Error	Error	Account balance cannot be less than 0	As expected
		account with positive balance (not "Account balance cannot be less than 0")	R4T2	WDR 1234567 1000 0000000 name	1234567 1000 name	1234567 000 name	1234567	Success	As expected
	are between 3 and 30 alphanumeric characters, possibly including spaces, but not beginning or ending with a space	name less than 3 digits	R5T1	WDR 1234567 1000 0000000 name	1234567 1000 na	Error	Error	Name not between 3 and 30 chars	As expected
		name more than 30 digits	R5T2	WDR 1234567 1000 0000000 name	1234567 1000 namenamenamenamenamenamenamename	Error	Error	Name not between 3 and 30 chars	As expected
		legal name	R5T5	WDR 1234567 1000 0000000 name	1234567 1000 name	1234567 000 name	1234567	Success	As expected
	monetary amounts are between 3 and 8 decimal digits, 000 to 99999999	monetary is less than 3 digits	R6T1	WDR 1234567 10 0000000 name	1234567 18 name	Error	Error	Monetary not between 3 and 8 digits	As expected
		monetary is more than 8 digits	R6T2	WDR 1234567 1000 0000000 name	1234567 10000000 name	Error	Error	Monetary not between 3 and 8 digits	As expected
		legal monetary	R6T3	WDR 1234567 1000 0000000 name	1234567 1000 name	1234567 000 name	1234567	Success	As expected

Descriptions on how we modified the template to conduct the testing

To conduct the testing, we have utilized the CI-Java-Maven-Template and its *runAndTest* helper function to test for different cases. We then ran the test script through Maven.

Within the ***runAndTest*** helper function, there are five parameters:

1. **The first parameter is :** A list of string as stored ion old master file
   - For example: "1234567 1000 joseph" denotes one account with account number 1234567 that has a balance of 1000 and a name of "joseph"
2. **The second parameter is:** A list of transactions produced by the front-end
   - For example: "DEP 1234567 1000 0000000 ***"
3. **The third parameter is unchanged:** A list of string expected at the tail of terminal output
4. **The fourth parameter is:** A list of string expected to be in the output of the new master file
   - For example: "Error" denotes error occurred while running the program, and "1234567 1000 joseph" denotes a new master file
5. **The fifth parameter is:** A list of string expected to be in the output of the new valid account list
   - For example: "1234567 1234568" denotes two accounts in the output valid account list, and "Error" denotes error occurred.

We then modified the code within the runAndTest function in order to create two temp files for master account file and valid account list. Eventually, our program has three loops that contains assert to check equality.

Finally, for the sake of conducting the testing better, we modified the tests a bit by eliminating some unreasonable ones, especially ones that has unreasonable transaction summary statement since our front end is fully tested so such error will never occur, so it fits out program better.

<u>Documentation explaining the script and testing process</u>

The ***AppTest.java*** is a testing program that contains the implementation of *all* of our test cases.

You may notice that there are lots of methods within this program. The following documentation's sake is to guide you through them.

The methods that are under the "`//Create Account`" comment correspond to the test cases for the *Login* operation.

- Each method (e.g. R1T1) corresponds to one of the corresponding tests (e.g. CR1T1)
- The naming convention of the method is: "c" (shorthand for create) followed by the Test No. (e.g. R1T1) in the table.
- We have applied the method of input partition to test our program respectively.

The methods that are under the "`//Withdraw`" comment correspond to the test cases for the *Logout* operation.

- Each method (e.g. R1T1) corresponds to one of the corresponding tests (e.g. WR1T1)
- The naming convention of the method is: "w" (shorthand for withdraw) followed by the Test No. (e.g. R1T1) in the table.
- We have applied the method of output partition to test our program respectively.